Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή

στα Πληροφοριακά Συστήματα

Ανάπτυξη Υβριδικού Συστήματος με Χρήση Συσκευής FPGA για την Επεξεργασία Δεδομένων με Γλώσσα SQL

Ιωάννης Βενετικίδης ΑΜ. 10800373

Επιβλέπων Καθηγητές
Βασίλειος Βασσάλος, Αναπληρωτής Καθηγητής
Μιχάλης Ψαράκης, Επίκουρος Καθηγητής

Οκτώβριος 2012

Δομή μεταπτυχιακής διατριβής

- FPGA – VHDL
- Σχεσιακή Άλγεβρα – SQL – Επέκταση Επερωτήσεων SQL σε VHDL
- Ανάγνωση και Εγγραφή Αρχείου, CAM (CAM_ARRAY), RAM-ROM, RS232 - UART
- Πειραματικά αποτελέσματα

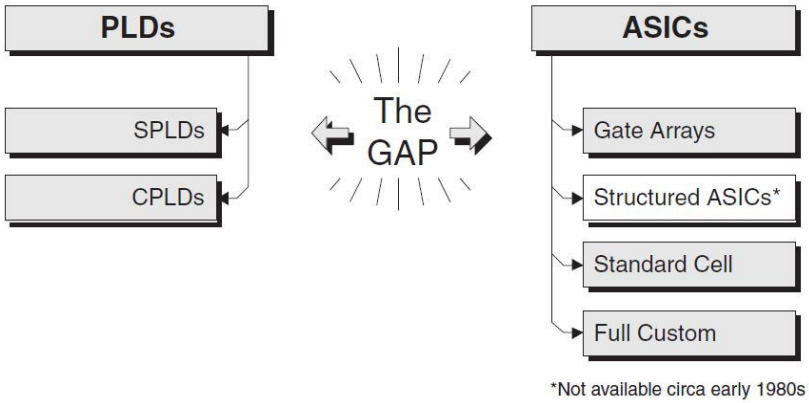
Στόχοι Μεταπτυχιακής Διατριβής

- Να αναδείχθεί η χρήση των συσκευών FPGA ως ένα εργαλείο συν-επεξεργασίας δεδομένων για άλλα μεγαλύτερα πληροφοριακά συστήματα, αλλά και ως ανεξάρτητη συσκευή επεξεργασίας δεδομένων.

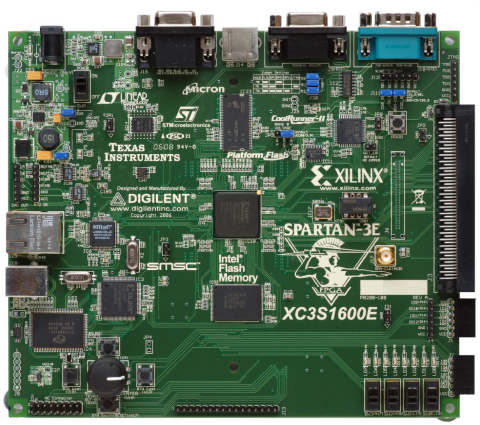
- Να καθορίστεί ο τρόπος και μεθοδολογία σχεδίασης τέτοιων κυκλωμάτων με πολύ αναλυτικά βήματα, καθιστώντας την μεταπτυχιακή διατριβή έναν πλήρη οδηγό χρήσης για μελλοντικές αντίστοιχες προσπάθειες.

- Η έρευνα που υλοποιήθηκε, να συμβάλει στην εξέλιξη των μη παραδοσιακών τρόπων επεξεργασία δεδομένων.

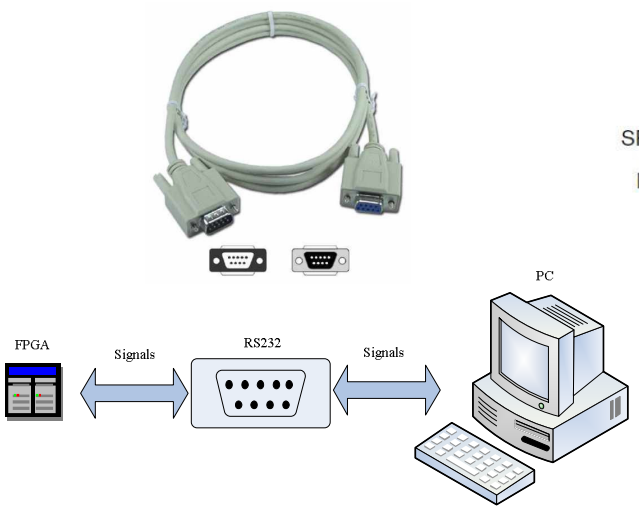
FPGA

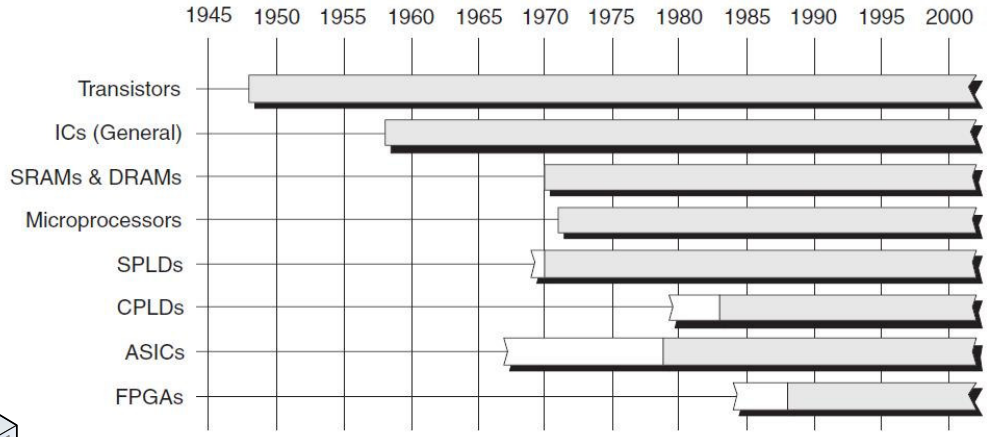


[Max] Το χάσμα μεταξύ διατάξεων PLDs και ASICs



[KitS] Spartan-3E 1600E





[Max] Χρονική εξέλιξη της τεχνολογίας των ολοκληρωμένων κυκλωμάτων, (οι ημερομηνίες είναι κατά προσέγγιση)

Επερωτήσεις **Q1,Q2,Q3** και **Q4**

R

A_1	A_2	A_3
1	4	5
7	9	8
5	3	5
4	2	1
2	7	4
1	2	2

S

B_1	B_2	B_3
2	4	8
1	1	1
2	5	5
1	2	1
2	5	4
1	2	3
4	6	9
8	4	2
1	1	3
3	7	2
5	3	1

Q1: SELECT A_1, A_2, A_3
FROM R
WHERE $A_3 > '2'$

Q1

A_1	A_2	A_3
1	4	5
7	9	8
5	3	5
2	7	4

Αποτέλεσμα

π_{A_1,A_2,A_3}

σ_a

$>_{a:(A_3,2)}$

R

Q2: SELECT B_1, B_2, B_3
FROM S
WHERE $B_1 = '1'$ AND $B_2 > '1'$

Q2

B_1	B_2	B_3
1	2	1
1	2	3

Αποτέλεσμα

π_{B_1,B_2,B_3}

σ_c

$\wedge_{c:(a,b)}$

$=_{a:(B_1,1)}$

$>_{b:(B_2,1)}$

S

Q3: SELECT $A_1, A_2, A_3, B_1, B_2, B_3$
FROM R, S
WHERE $A_1 > '2'$ AND $A_2 = B_2$

Q3

A_1	A_2	A_3	B_1	B_2	B_3
5	3	5	5	3	1
4	2	1	1	2	1
4	2	1	1	2	3

Αποτέλεσμα

$\pi_{A_1,A_2,A_3,B_1,B_2,B_3}$

σ_c

$\wedge_{c:(a,b)}$

$>_{a:(A_1,2)}$

$=_{b:(A_2,B_2)}$

R,S

Q4: SELECT A_1, A_2, A_3, B_2, B_3
FROM R, S
WHERE $A_1 = B_1$

Q4

A_1	A_2	A_3	B_2	B_3
1	4	5	1	1
1	4	5	2	1
1	4	5	2	3
1	4	5	1	3
5	3	5	3	1
4	2	1	6	9
2	7	4	4	8
2	7	4	5	5
2	7	4	5	4
1	2	2	1	1
1	2	2	2	1
1	2	2	2	3
1	2	2	1	3

Αποτέλεσμα

$\pi_{A_1,A_2,A_3,B_2,B_3}$

σ_c

$=_{a:(A_1,B_1)}$

R,S

Επέκταση Επερώτησης Q1 σε VHDL

```vhdl
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.STD_LOGIC_ARITH.ALL;
23  use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25  ---- Uncomment the following library declaration if instantiating
26  ---- any Xilinx primitives in this code.
27  --library UNISIM;
28  --use UNISIM.VComponents.all;
29
30  entity Q1_select is
31
32  Port (
33          clk        : in  std_logic;                        -- clock input
34          rst        : in  std_logic;                        -- reset input
35          in_data    : in  STD_LOGIC_VECTOR (23 downto 0);   -- data input
36          valid_bit  : out  STD_LOGIC;                       -- valid bit output
37          out_data   : out  STD_LOGIC_VECTOR (23 downto 0)   -- data output
38          );
39  end Q1_select;
40
41  architecture Behavioral of Q1_select is
42
43  -- control signals
44  signal A1      :  std_logic_vector(7 downto 0);      -- the first element A1 of the tuple
45  signal A2      :  std_logic_vector(7 downto 0);      -- the second element A2 of the tuple
46  signal A3      :  std_logic_vector(7 downto 0);      -- the third element A3 of the tuple
47  signal s_in_data :  STD_LOGIC_VECTOR (23 downto 0); -- the signal from input data
48
49  begin
50
51  -- signals  initialization
52  s_in_data <= in_data;
53  A1 <= s_in_data (23 downto 16);      -- initialize first element A1 of the tuple
54  A2 <= s_in_data (15 downto  8);      -- initialize second element A2 of the tuple
55  A3 <= s_in_data ( 7 downto  0);      -- initialize third element A3 of the tuple
56
57  ---------------------------------------------- Q1 ----------------------------------------------
58  Q1: process (clk)                    -- start process Q1
59     begin
60     if clk'event and clk='1' then     -- synchronization whit clock
61        if (A3 > "00000010") then      -- the control element A3 > 2
62           out_data <= s_in_data;      -- output correct data
63           valid_bit <= '1' ;          -- valid bit = '1'
64        else
65           out_data <= (others =>'0'); -- output null data
66           valid_bit <= '0' ;          -- valid bit = '0'
67           end if;
68     end if;
69  end process Q1;                      --end process Q1
70
71  end Behavioral;
```

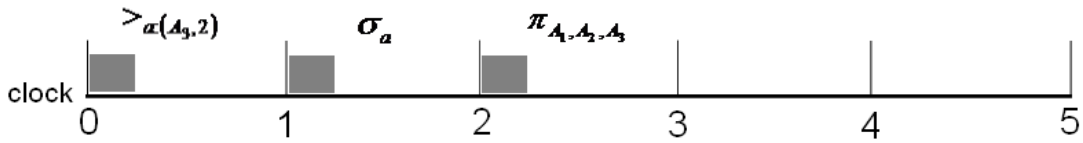
Κώδικας VHDL που υλοποιεί την επερώτηση Q1

Q1: SELECT A_1, A_2, A_3

FROM R

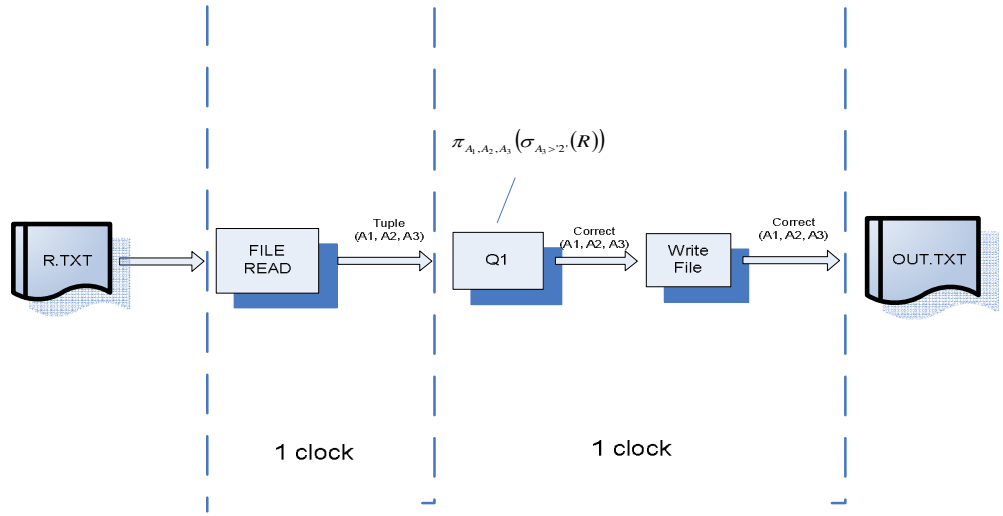
WHERE $A_3 > '2'$

$$\pi_{A_1, A_2, A_3}\left(\sigma_{A_3 > '2'}(R)\right)$$



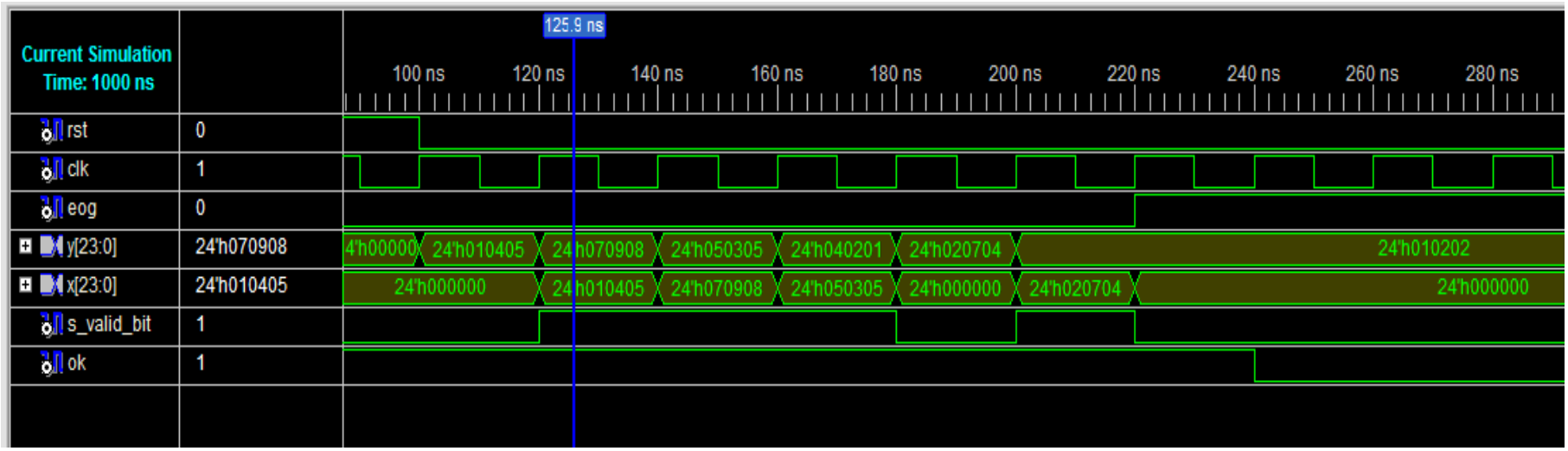
Συμβατική χρονική ακολουθία επερώτησης Q1 από ένα ΣΒΔ, ανεξαρτήτου επεξεργαστικής ταχύτητας

Χρονική ακολουθία επερώτησης Q1 μετά από σχεδίαση στη γλώσσα VHDL και εκτέλεση του στη συσκευή FPGA



Σχεδιαστική πρόταση που υλοποιεί την επίλυση της επερώτησης Q1

Πειραματικά Αποτελέσματα Επερώτησης Q1



Προσομοίωση λογικού κυκλώματος επερώτησης Q1

R.txt

```
000000010000010000000101
000001110000100100001000
000001010000001100000101
000001000000001000000001
000000100000011100000100
000000010000001000000010
```

R.txt *

```
00000001 00000100 00000101
00000111 00001001 00001000
00000101 00000011 00000101
00000100 00000010 00000001
00000010 00000111 00000100
00000001 00000010 00000010
```

A1 A2 A3

Εμφάνιση περιεχομένων αρχείου R.txt, για χρήση του κατά την προσομοίωση της επερώτησης Q1

OUT.TXT

```
000000010000010000000101
000001110000100100001000
000001010000001100000101
000000100000011100000100
```

OUT.TXT *

```
00000001 00000100 00000101
00000111 00001001 00001000
00000101 00000011 00000101
00000010 00000111 00000100
```

A1 A2 A3

Εμφάνιση περιεχομένων αρχείου OUT.txt κατά την προσομοίωση της επερώτησης Q1

R

A_1	A_2	A_3
1	4	5
7	9	8
5	3	5
4	2	1
2	7	4
1	2	2

Q1: SELECT A_1, A_2, A_3
FROM R
WHERE $A_3 > '2'$

Q1

A_1	A_2	A_3
1	4	5
7	9	8
5	3	5
2	7	4

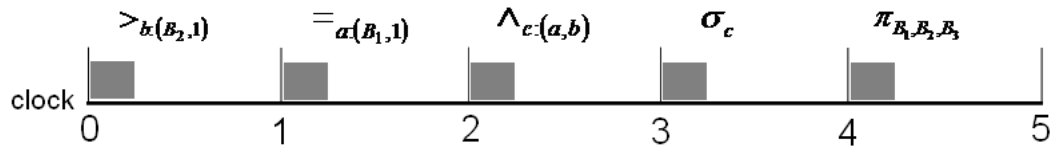
Επέκταση Επερώτησης Q2 σε VHDL

```vhdl
54  begin
55  -- signals  initialization
56  s_in_data <= in_data;
57  B1 <= s_in_data (23 downto 16);     -- initialize first element B1 of the tuple
58  B2 <= s_in_data (15 downto  8);     -- initialize second element B2 of the tuple
59  B3 <= s_in_data ( 7 downto  0);     -- initialize third element B3 of the tuple
60
61  ---------------------------------------------- Q2_A ----------------------------------------------
62  Q2_A: process (clk)                 -- start process Q2_A
63    begin
64    if clk'event and clk='1' then     -- synchronization whit clock
65      if (B1 = "00000001") then       -- the control element B1 = '1'
66        valid_bit_A <= '1' ;          -- valid bit A = '1'
67        s_in_data_B <= in_data;       -- correct data
68      else
69        valid_bit_A <= '0' ;          -- valid bit A = '0'
70        end if;
71    end if;
72  end process Q2_A;                   -- end process Q2_A
73
74  ---------------------------------------------- Q2_B ----------------------------------------------
75  Q2_B: process (clk)                 -- start process Q2_B
76    begin
77    if clk'event and clk='1' then     -- synchronization whit clock
78      if (B2 > "00000001") then       -- the control element B2 > 1
79        valid_bit_B <= '1' ;          -- valid bit B = '1'
80        s_in_data_B <= in_data;       -- correct data
81      else
82        valid_bit_B <= '0' ;          -- valid bit B = '0'
83        end if;
84    end if;
85  end process Q2_B;                   -- end process Q2_B
86
87  ---------------------------------------------- Q2_C ----------------------------------------------
88  Q2_C:process (clk,valid_bit_A,valid_bit_B)   -- start process Q2_C
89    begin
90    -- if clk'event and clk='1' then          -- synchronization whit clock
91        if (valid_bit_A = '1' and valid_bit_B = '1') then -- if the comparisons are correct
92          out_data <= s_in_data_B;      -- output correct data
93          valid_bit <= '1' ;            -- valid bit = '1'
94        else
95          out_data <= (others =>'0');   -- output null data
96          valid_bit <= '0' ;            -- valid bit = '0'
97        end if;
98    -- end if;
99  end process Q2_C;                       -- end process Q2_C
100
101 end Behavioral;
```

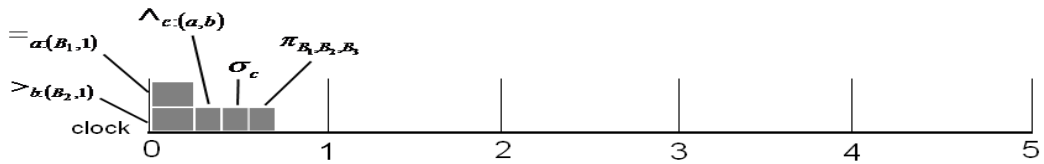
Κώδικας VHDL που υλοποιεί την επερώτηση Q2

Q2: SELECT B_1, B_2, B_3
FROM S
WHERE $B_1 = '1'$ AND $B_2 > '1'$

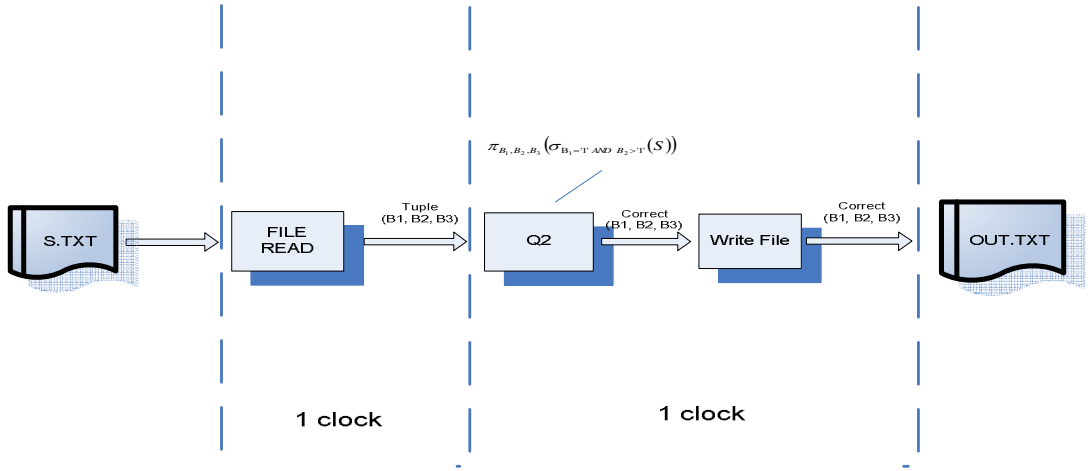
$$\pi_{B_1,B_2,B_3}\left(\sigma_{B_1='1'\ AND\ B_2>'1'}(S)\right)$$



Συμβατική χρονική ακολουθία επερώτησης Q2 από ένα ΣΒΔ, ανεξαρτήτου επεξεργαστικής ταχύτητας

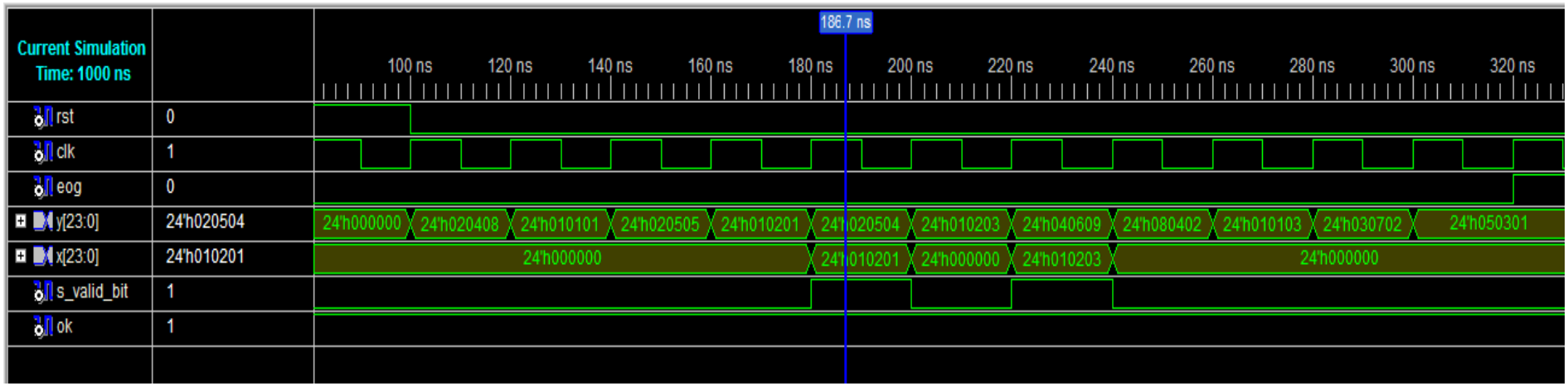


Χρονική ακολουθία επερώτησης Q2 μετά από σχεδίαση στη γλώσσα VHDL και εκτέλεση του στη συσκευή FPGA

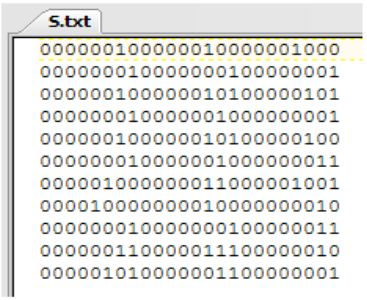


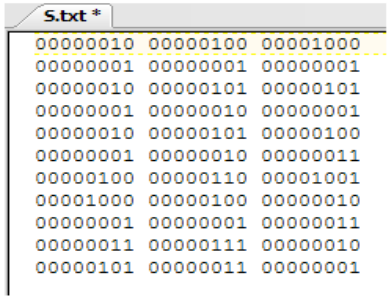
Σχεδιαστική πρόταση που υλοποιεί την επίλυση της επερώτησης Q2

Πειραματικά Αποτελέσματα Επερώτησης Q2



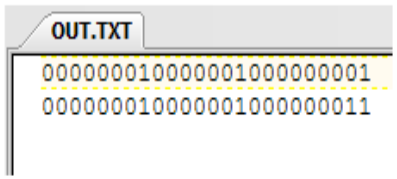
Προσομοίωση λογικού κυκλώματος επερώτησης Q2

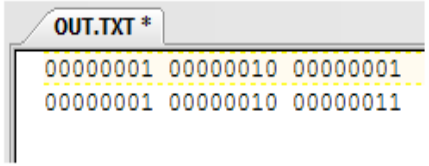




B1 B2 B3

Εμφάνιση περιεχομένων αρχείου S.txt, για χρήση του κατά την προσομοίωση της επερώτησης Q2





B1 B2 B3

Εμφάνιση περιεχομένων αρχείου OUT.txt κατά την προσομοίωση της επερώτησης Q2

S

B_1	B_2	B_3
2	4	8
1	1	1
2	5	5
1	2	1
2	5	4
1	2	3
4	6	9
8	4	2
1	1	3
3	7	2
5	3	1

→

Q2: SELECT B_1, B_2, B_3

FROM S

WHERE B_1 = '1' AND B_2 > '1'

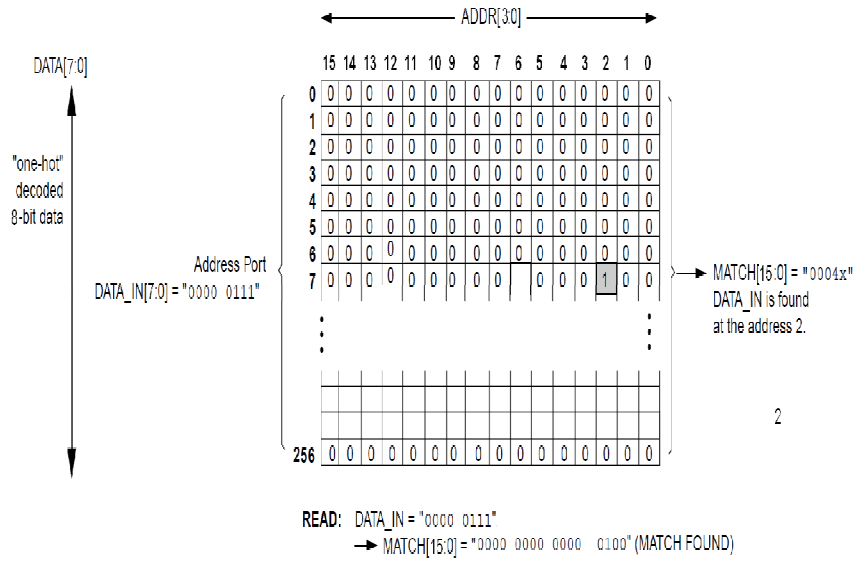
→

Q2

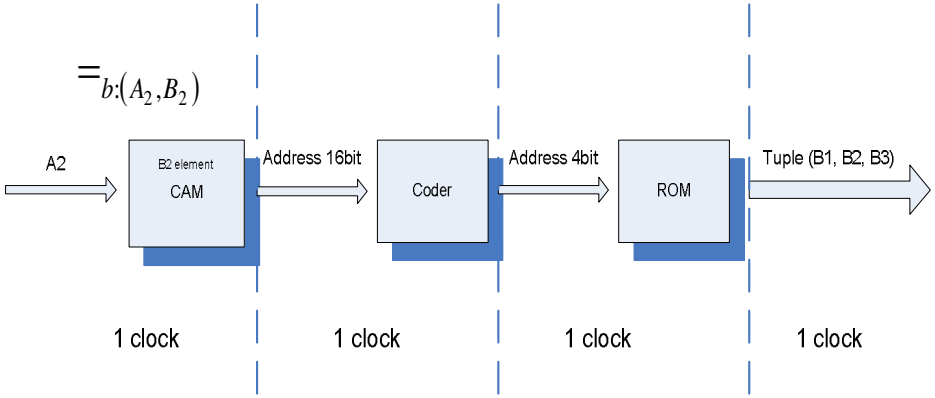
B_1	B_2	B_3
1	2	1
1	2	3

Μνήμη CAM (Contact Addressable Memory)

- Επιτελεί την αντίστροφη λειτουργία από αυτή που επιτελούν οι παραδοσιακές μνήμες.
- Το στοιχείο αναζήτησης μέσα στη μνήμη δεν είναι η διεύθυνση αλλά το περιεχόμενο της, δηλαδή τα δεδομένα.
- Ο χρόνος απόκρισης της μνήμης στην αναζήτηση είναι της τάξεως του ενός κύκλου ρολογιού (!), για κάθε προσπέλαση.
- Μια μνήμη CAM σε συνδυασμό με μια μνήμη RAM – ROM κάνει το χρόνος απόκρισης (για την εύρεση δεδομένων μέσα στη μνήμη RAM – ROM) να φτάνει τους δυο με τρεις κύκλους ρολογιού.
- Αν περιέχει περισσότερα από ένα ίδια στοιχεία μέσα στη μνήμη, τότε ο χρήστης έχει τη δυνατότητα να λάβει την πρώτη ή την τελευταία θέση της.
- Αν υπάρχουν περισσότερες από μία *επιτυχίες εύρεσης* (*hit*) μπορεί να επιλέξει να εμφανίζονται όλες αυτές υπό την μορφή bit.
- Αν χρειαστεί να μην υπάρχουν ίδια στοιχεία μέσα στην μνήμη CAM, θα πρέπει να δημιουργηθούν ξεχωριστά λογικά κυκλώματα που το καθένα θα περιέχει διαφορετικές CAM αλλά και RAM – ROM.



Δομή περιεχομένων μνήμης CAM και ο τρόπος εύρεσης στοιχείου



Διαδικασία εκμαιεύσεις δεδομένων από τη μνήμη της επερώτησης Q3

Περιεχόμενα Μνήμών CAM και ROM Επερώτησης Q3 , Αλγόριθμος *CAM_ARRAY*

```
Problems  Tasks  Properties  Console
<terminated> CAM_Array.exe [C/C++ Application] D:\eClipse\CAM_Array\Debug\CAM_Array.exe (22/8/12 6:09 μμ.)

Ο πίνακας εισαγωγής V: { 4, 1, 5, 2, 5, 2, 6, 4, 1, 7, 3,}

Ο ταξινομημένος πίνακας εισαγωγής V: { 1, 1, 2, 2, 3, 4, 4, 5, 5, 6, 7,}

Ο αριθμός των πινάκων που χρειαζόμαστε για την δημιουργία των μνημών CAM είναι 2  και το βάθος αυτών είναι 6 εγγραφές για το καθένα

Ο πίνακας V πλέον είναι μηδενικός

Χρειαζόμαστε λοιπόν τα παρακάτω:

        1ος πίνακας : { 5, 4, 2, 1, 7, 3,}

        2ος πίνακας : { 5, 4, 2, 1, 6, 0,}
```

Εκτέλεση του αλγόριθμου CAM_ARRAY στο γνώρισμα B2 του πίνακα S, της επερώτησης Q3

CAM A

B_2
4
1
5
2
7
3

ROM A

B_1	B_2	B_3
2	4	8
1	1	1
2	5	5
1	2	1
3	7	2
5	3	1

CAM B

B_2
5
2
6
4
1

ROM B

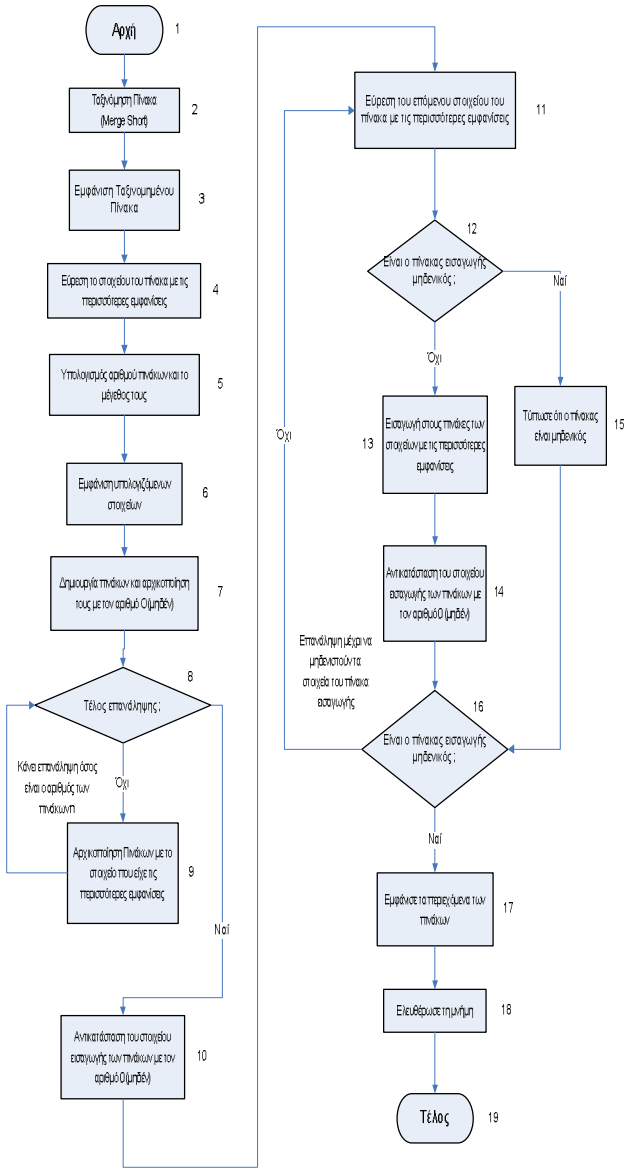
B_1	B_2	B_3
2	5	4
1	2	3
4	6	9
8	4	2
1	1	3

Τα περιεχόμενα κάθε μνήμης της επερώτησης Q3

B_1	B_2	B_3	ROM
2	4	8	romA
1	1	1	romA
2	5	5	romA
1	2	1	romA
2	5	4	romB
1	2	3	romB
4	6	9	romB
8	4	2	romB
1	1	3	romB
3	7	2	romA
5	3	1	romA

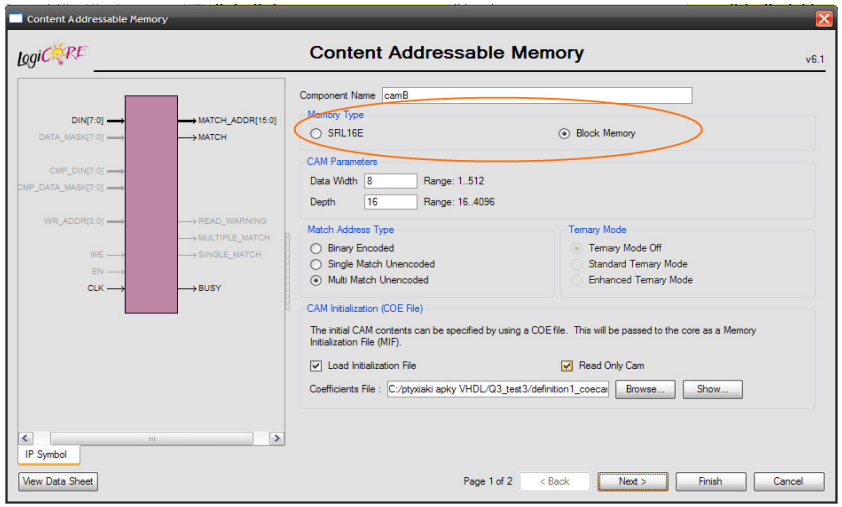
B_2	CAM
4	camA
1	camA
5	camA
2	camA
5	camB
2	camB
6	camB
4	camB
1	camB
7	camA
3	camA

Κατανομή στοιχείων πίνακα S στις αντίστοιχες μνήμες CAM και ROM, της επερώτησης Q3

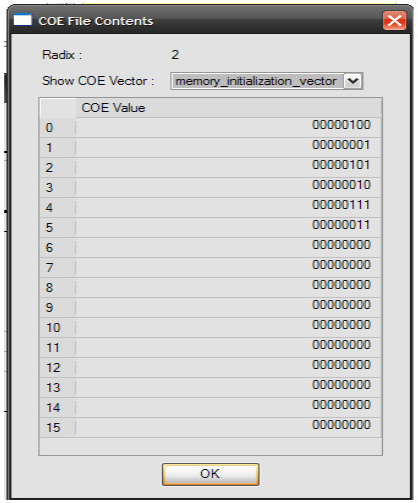


Τα βήματα και η δομή του αλγόριθμου CAM_ARRAY, το διάγραμμα ροής του

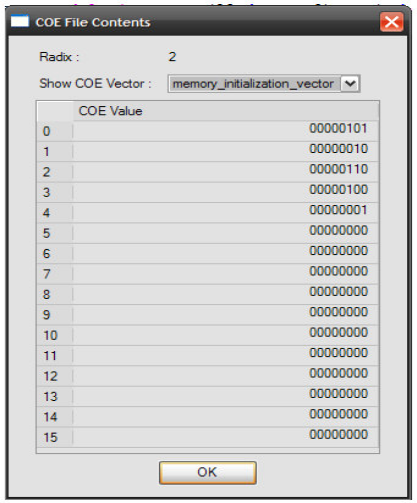
CAM, RAM – ROM και αρχικοποίηση τους, επερώτηση Q3



Τύπος μνήμης Block Memory επερωτήσεων Q3 και Q4 στο εργαλειο σχεδιασης Xilinx ISE

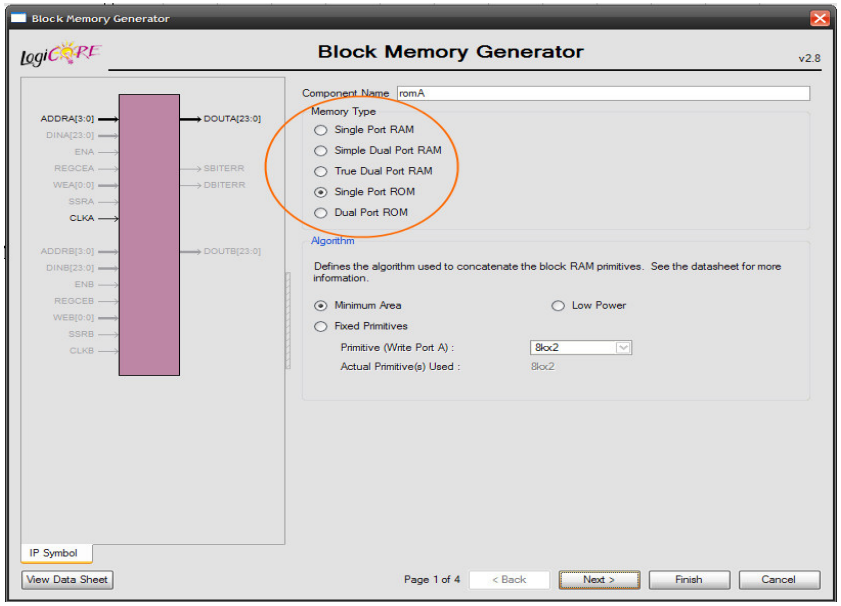


camA

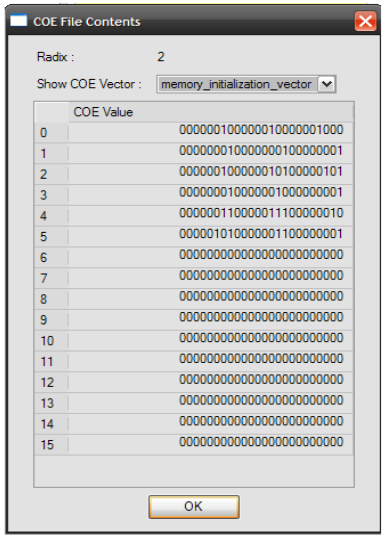


camB

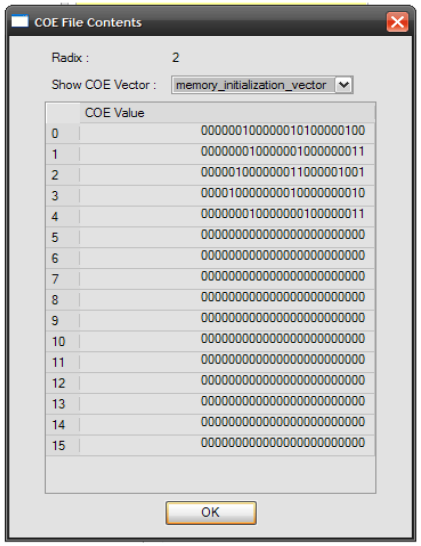
Εμφάνιση περιεχομένων μνημών camA και camB επερώτησης Q3



Επιλογή μνήμης Single Port ROM για τη διευθέτηση της μνήμης romA της επερώτησης Q3, από το εργαλειο σχεδιασης Xilinx ISE



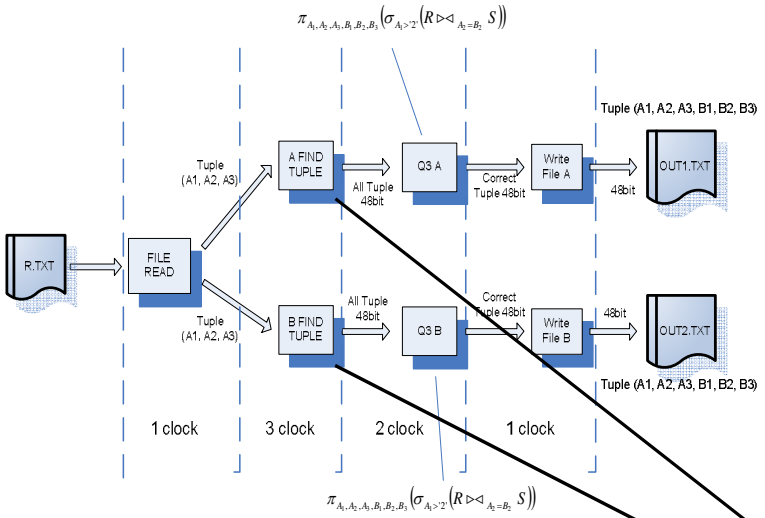
romA



romB

Εμφάνιση περιεχομένων μνημών romA και romB επερώτησης Q3

Επέκταση Επερώτησης Q3 σε VHDL



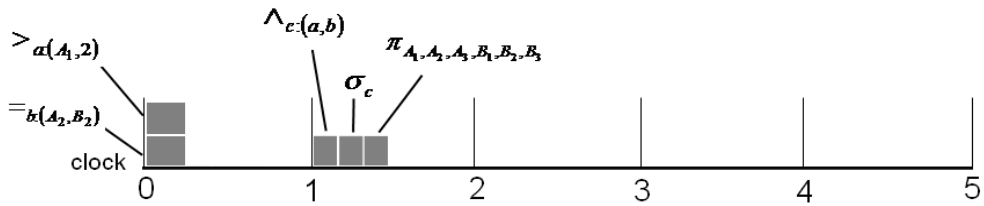
Αρχιτεκτονική δομή λογικού κυκλώματος που δίνει λύση στην επίλυση ερωτημάτων της επερώτησης Q3

Q3: SELECT $A_1, A_2, A_3, B_1, B_2, B_3$

FROM R, S

WHERE $A_1 > '2'$ AND $A_2 = B_2$

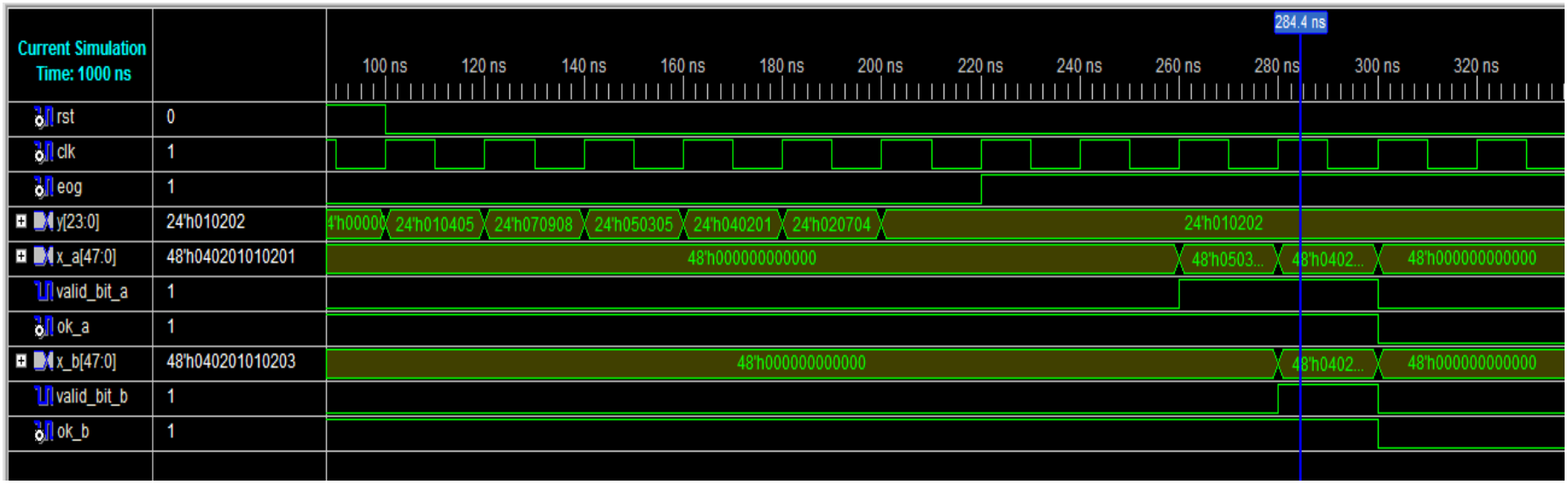
$$\pi_{A_1,A_2,A_3,B_1,B_2,B_3}(\sigma_{A_1>'2'}(R \bowtie_{A_2=B_2} S))$$



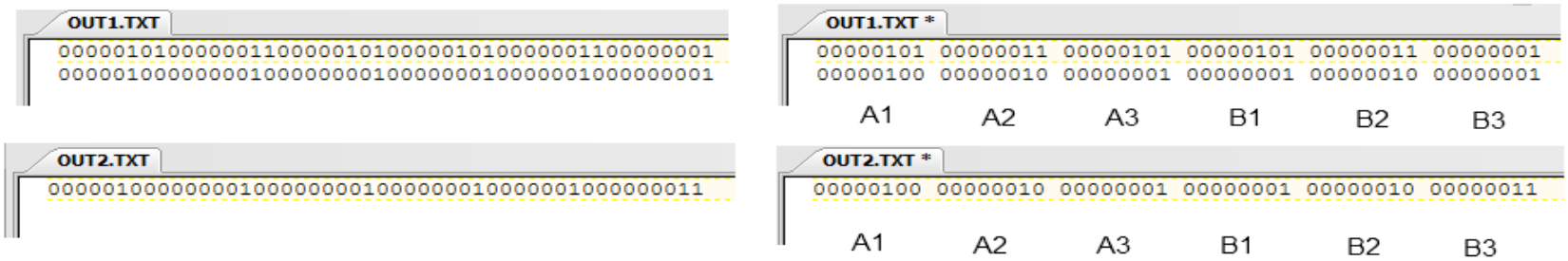
Χρονική ακολουθία επερώτησης Q3 μετά από σχεδίαση στη γλώσσα VHDL και εκτέλεση του στη συσκευή FPGA

```
begin
-- signals  initialization
s_in_data <= tuple_A_and_B;
A1 <= s_in_data (47 downto 40);      -- initialize first element A1 of the tuple
A2 <= s_in_data (39 downto 32);      -- initialize second element A2 of the tuple
A3 <= s_in_data (31 downto 24);      -- initialize third element A3 of the tuple
B1 <= s_in_data (23 downto 16);      -- initialize first element B1 of the tuple
B2 <= s_in_data (15 downto  8);      -- initialize second element B2 of the tuple
B3 <= s_in_data ( 7 downto  0);      -- initialize third element B3 of the tuple
------------------
```

Πειραματικά Αποτελέσματα Επερώτησης Q3



Προσομοίωση λογικού κυκλώματος επερώτησης Q3



Εμφάνιση περιεχομένων αρχείου OUT1.txt και OUT2.txt κατά την προσομοίωση της επερώτησης Q3

R

A_1	A_2	A_3
1	4	5
7	9	8
5	3	5
4	2	1
2	7	4
1	2	2

Q3: SELECT $A_1, A_2, A_3, B_1, B_2, B_3$
FROM R, S
WHERE $A_1 > '2'$ AND $A_2 = B_2$

S

B_1	B_2	B_3
2	4	8
1	1	1
2	5	5
1	2	1
2	5	4
1	2	3
4	6	9
8	4	2
1	1	3
3	7	2
5	3	1

Q3

A_1	A_2	A_3	B_1	B_2	B_3
5	3	5	5	3	1
4	2	1	1	2	1
4	2	1	1	2	3

Μνήμες CAM και ROM Επερώτησης Q4

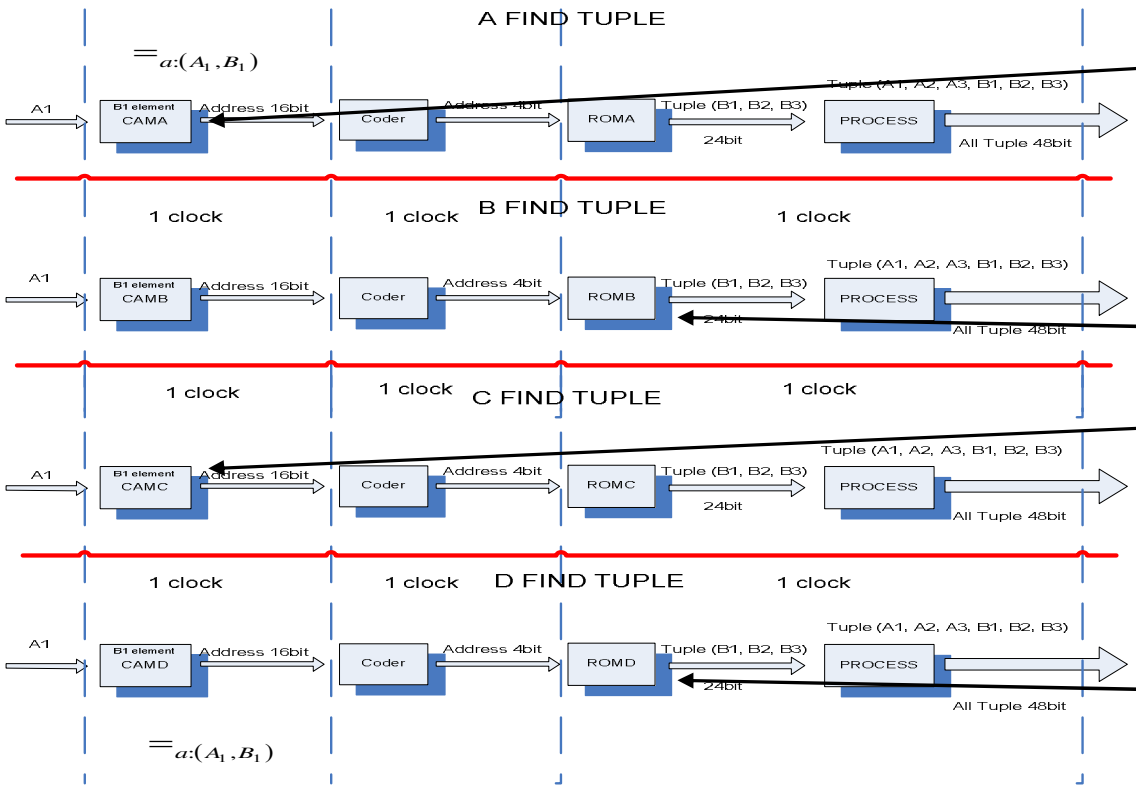
```
Problems  Tasks  Properties  Console
<terminated> CAM_Array.exe [C/C++ Application] D:\eclipse\CAM_Array\Debug\CAM_Array.exe (25/8/12 12:25 π.μ.)

Ο πίνακας εισαγωγής V: { 2, 1, 2, 1, 2, 1, 4, 8, 1, 3, 5,}

Ο ταξινομημένος πίνακας εισαγωγής V: { 1, 1, 1, 1, 2, 2, 2, 3, 4, 5, 8,}

Ο αριθμός των πινάκων που χρειαζόμαστε για την δημιουργία των μνημών CAM είναι 4  και το βάθος αυτών είναι 3 εγγραφές για το καθένα

Ο πίνακας V πλέον είναι μηδενικός

Χρειαζόμαστε λοιπόν τα παρακάτω:

        1ος πίνακας : { 1, 2, 5,}

        2ος πίνακας : { 1, 2, 4,}

        3ος πίνακας : { 1, 2, 3,}

        4ος πίνακας : { 1, 8, 0,}
```

Εκτέλεση του αλγόριθμου CAM_ARRAY στο γνώρισμα B1του πίνακα S, της επερώτησης Q4

B_1	B_2	B_3	ROM
2	4	8	romA
1	1	1	romA
2	5	5	romB
1	2	1	romB
2	5	4	romC
1	2	3	romC
4	6	9	romB
8	4	2	romD
1	1	3	romD
3	7	2	romC
5	3	1	romA

B_1	CAM
4	romA
1	romA
5	romB
2	romB
5	romC
2	romC
6	romB
4	romD
1	romD
7	romC
3	romA

Κατανομή στοιχείων πίνακα S στις αντίστοιχες μνήμες CAM και ROM



Οι λογικές μονάδες FIND TUPLE και η λογική δομή τους, της επερώτησης Q4

CAM A

B_1
2
1
5

ROM A

B_1	B_2	B_3
2	4	8
1	1	1
5	3	1

CAM B

B_1
2
1
4

ROM B

B_1	B_2	B_3
2	5	5
1	2	1
4	6	9

CAM C

B_1
2
1
3

ROM C

B_1	B_2	B_3
2	5	4
1	2	3
3	7	2

CAM D

B_1
8
1

ROM D

B_1	B_2	B_3
8	4	2
1	1	3

Τα περιεχόμενα κάθε μνήμης της επερώτησης Q4

Επέκταση Επερώτησης Q4 σε VHDL

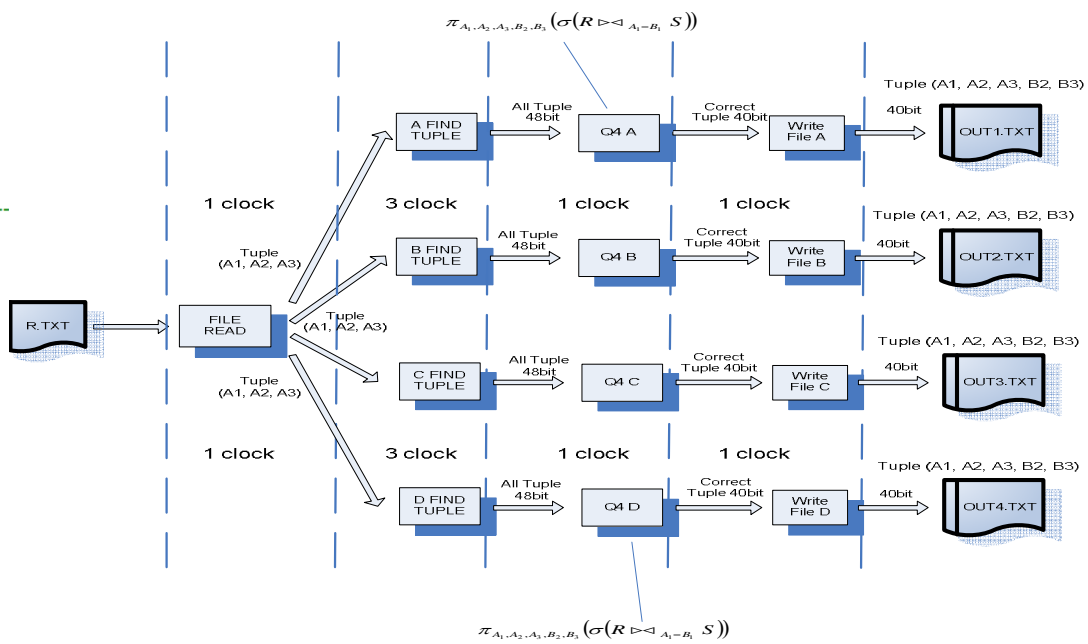
```
41
42  architecture Behavioral of Q4_select is
43
44  -- control signals
45  signal A1  :  std_logic_vector(7 downto 0);          -- the first element A1 of the tuple
46  signal A2  :  std_logic_vector(7 downto 0);          -- the second element A2 of the tuple
47  signal A3  :  std_logic_vector(7 downto 0);          -- the third element A3 of the tuple
48
49  signal B1  :  std_logic_vector(7 downto 0);          -- the first element B1 of the tuple
50  signal B2  :  std_logic_vector(7 downto 0);          -- the second element B2 of the tuple
51  signal B3  :  std_logic_vector(7 downto 0);          -- the third element B3 of the tuple
52
53  signal s_in_data :  STD_LOGIC_VECTOR (47 downto 0);  -- the signal from input data
54  signal data_out_A1_A2_A3_B2_B3  :  std_logic_vector(39 downto 0) := (others => '0');
55
56  begin
57  -- signals  initialization
58  s_in_data <= tuple_A_and_B;
59  A1 <= s_in_data (47 downto 40);     -- initialize first element A1 of the tuple
60  A2 <= s_in_data (39 downto  32);    -- initialize second element A2 of the tuple
61  A3 <= s_in_data (31 downto  24);    -- initialize third element A3 of the tuple
62  B1 <= s_in_data (23 downto 16);     -- initialize first element B1 of the tuple
63  B2 <= s_in_data (15 downto  8);     -- initialize second element B2 of the tuple
64  B3 <= s_in_data ( 7 downto  0);     -- initialize third element B3 of the tuple
65
66  -- initialize the correct data A1, A2, A3, B2, B3
67  data_out_A1_A2_A3_B2_B3 <= A1 & A2 & A3 & B2 & B3;
68
69  ---------------------------------------------- Q4----------------------------------------------
70  Q4: process (clk)                   -- start process Q4
71    begin
72    if (clk'event and clk='1') then   -- synchronization whit clock
73      if (A1 = B1) then               -- the control element A1 = B1
74        out_data <= data_out_A1_A2_A3_B2_B3;  -- output correct data
75          valid_bit <= '1' ;          -- valid bit = '1'
76      else
77        out_data <= (others =>'0');   -- output null data
78          valid_bit <= '0' ;          -- valid bit = '0'
79        end if;
80    end if;
81  end process Q4;                     -- end process Q4
82
83  end Behavioral;
84
```

Κώδικας VHDL που υλοποιεί την επερώτηση Q4

Q4: SELECT A_1, A_2, A_3, B_2, B_3
FROM R, S
WHERE $A_1 = B_1$

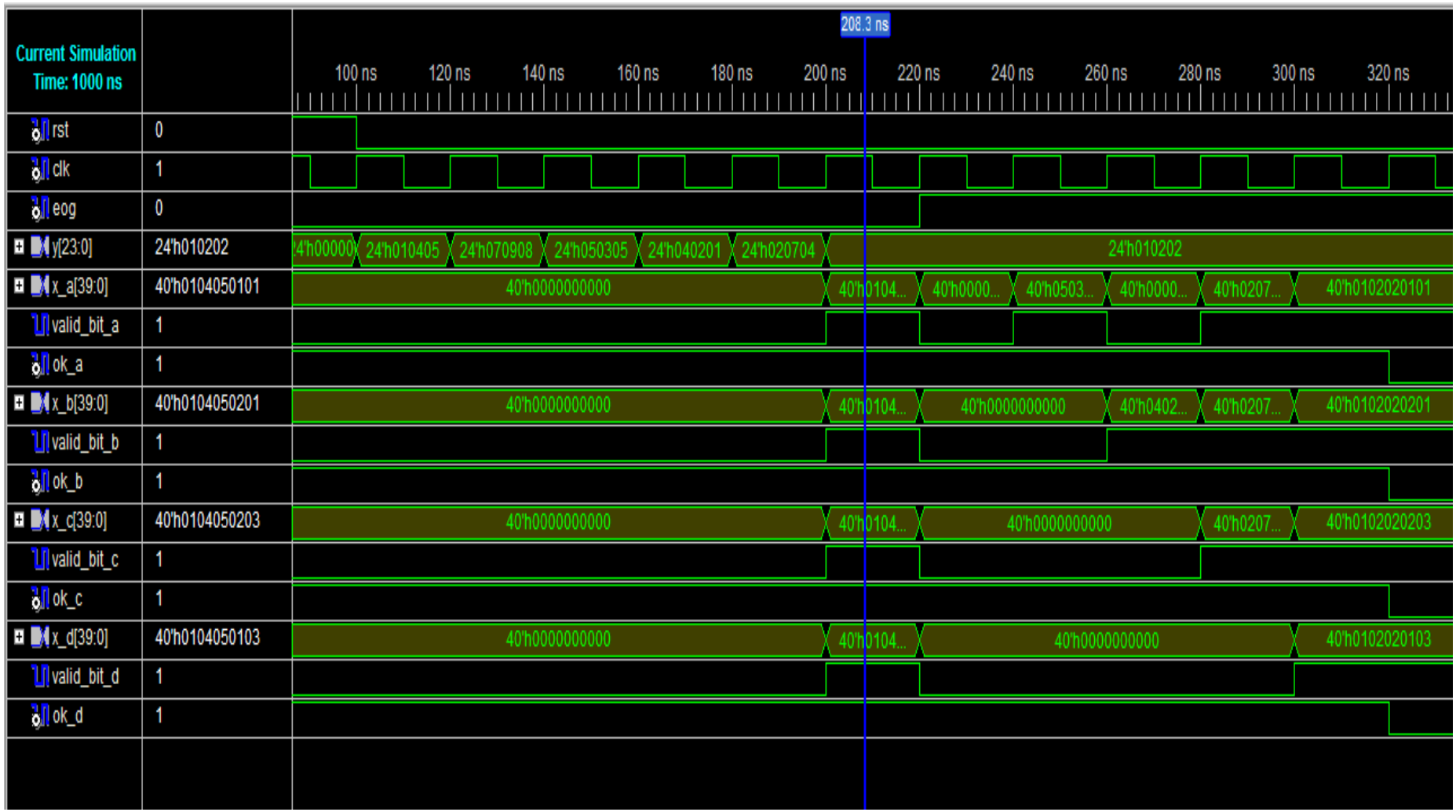
$$\pi_{A_1,A_2,A_3,B_2,B_3}\left(\sigma\left(R \bowtie_{A_1=B_1} S\right)\right)$$

Χρονική ακολουθία επερώτησης Q4 μετά από σχεδίαση στη γλώσσα VHDL και εκτέλεση του στη συσκευή FPGA



Αρχιτεκτονική δομή λογικού κυκλώματος που δίνει λύση στην επίλυση ερωτημάτων της επερώτησης Q4

Πειραματικά Αποτελέσματα Επερώτησης Q4



Προσομοίωση λογικού κυκλώματος επερώτησης Q4

Πειραματικά Αποτελέσματα Επερώτησης Q4

OUT1.TXT

```
00000001000001000000010100000001000000001
00000101000000110000010100000001100000001
00000010000001110000010000000010000001000
00000001000000100000000100000000100000001
```

OUT1.TXT *

A1	A2	A3	B2	B3
00000001	00000100	00000101	00000001	00000001
00000101	00000011	00000101	00000011	00000001
00000010	00000111	00000100	00000100	00001000
00000001	00000010	00000010	00000001	00000001

OUT2.TXT

```
00000001000001000000010100000001000000001
00000100000000010000000010000011000001001
00000010000001110000010000000010100000101
00000001000000100000000100000000100000001
```

OUT2.TXT *

A1	A2	A3	B2	B3
00000001	00000100	00000101	00000010	00000001
00000100	00000010	00000001	00000110	00001001
00000010	00000111	00000100	00000101	00000101
00000001	00000010	00000010	00000010	00000001

OUT3.TXT

```
00000001000001000000010100000001000000011
00000010000001110000010000000010100000100
00000001000000100000000100000000100000011
```

OUT3.TXT *

A1	A2	A3	B2	B3
00000001	00000100	00000101	00000010	00000011
00000010	00000111	00000100	00000101	00000100
00000001	00000010	00000010	00000010	00000011

OUT4.TXT

```
00000001000001000000010100000001000000011
00000001000000100000000100000000100000011
```

OUT4.TXT *

A1	A2	A3	B2	B3
00000001	00000100	00000101	00000001	00000011
00000001	00000010	00000010	00000001	00000011

Εμφάνιση περιεχομένων αρχείων OUT1.txt, OUT2.txt, OUT3.txt και OUT4.txt κατά την προσομοίωση της επερώτησης Q4

R

A_1	A_2	A_3
1	4	5
7	9	8
5	3	5
4	2	1
2	7	4
1	2	2

S

B_1	B_2	B_3
2	4	8
1	1	1
2	5	5
1	2	1
2	5	4
1	2	3
4	6	9
8	4	2
1	1	3
3	7	2
5	3	1

Q4: SELECT A_1, A_2, A_3, B_2, B_3

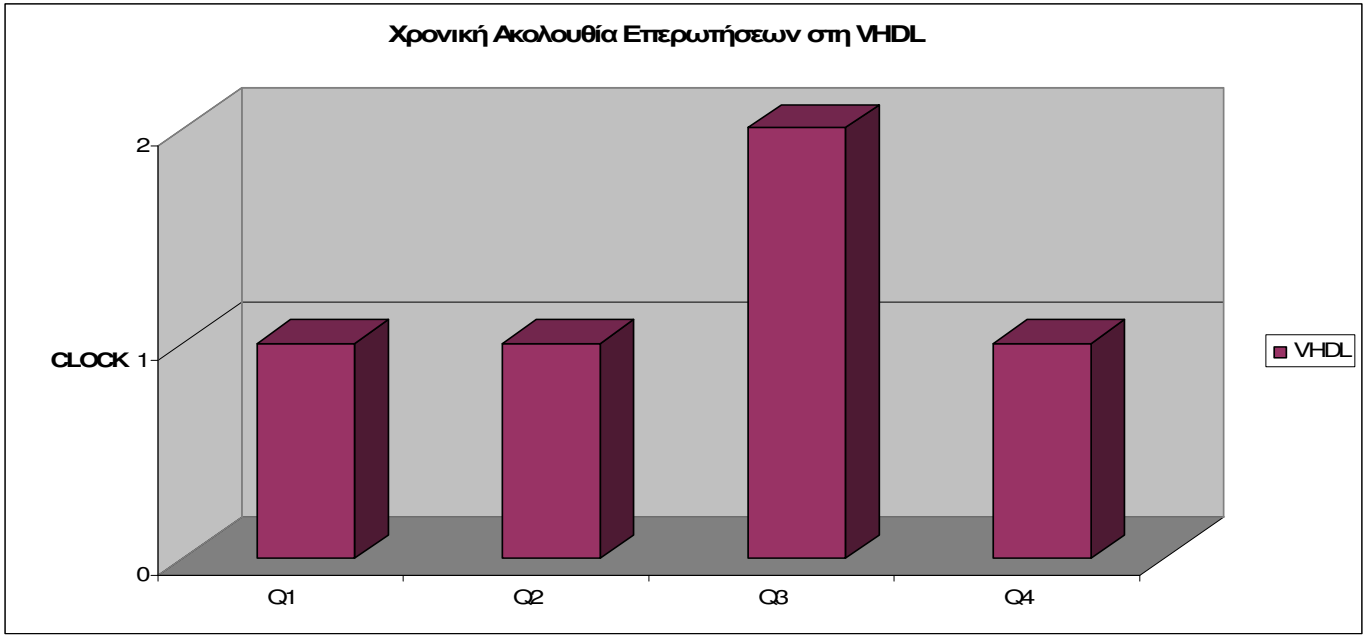
FROM R, S

WHERE $A_1 = B_1$

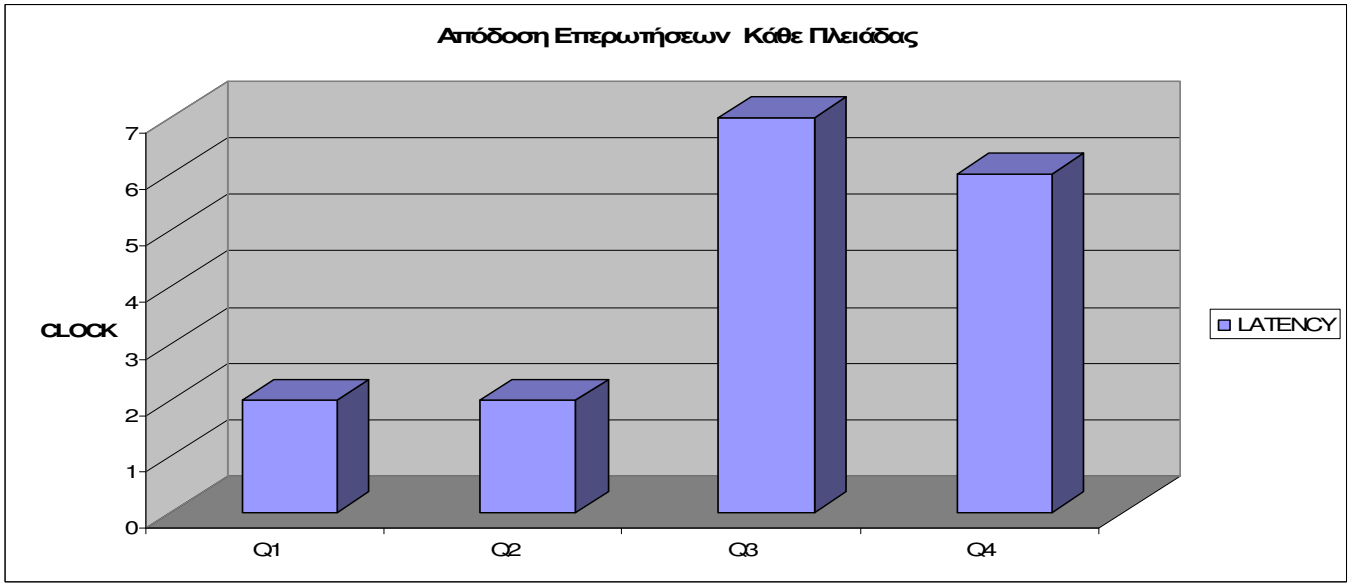
Q4

A_1	A_2	A_3	B_2	B_3
1	4	5	1	1
1	4	5	2	1
1	4	5	2	3
1	4	5	1	3
5	3	5	3	1
4	2	1	6	9
2	7	4	4	8
2	7	4	5	5
2	7	4	5	4
1	2	2	1	1
1	2	2	2	1
1	2	2	2	3
1	2	2	1	3

Εκτίμηση Αποτελεσμάτων



Χρονική ακολουθία επερώτησης μετά από σχεδίαση στη γλώσσα ***VHDL*** και εκτέλεση του στη συσκευή FPGA



Ο ***χρόνος εκτέλεσης*** (***Latency***) είναι ο χρόνος διευθέτησης των ερωτημάτων, δηλαδή ο χρόνος που απαιτείται για την επεξεργασία των δεδομένων από τη στιγμή που επιλέγονται και μπαίνουν στην επεξεργαστική μηχανή, (από την πιο απλή μέχρι την πιο πολύπλοκη) μέχρι την έξοδο από αυτήν.

Μελλοντικές Επεκτάσεις

- **Υλοποίηση στο υλικό**, να εκτελεστεί σίγουρα η επέκταση των επερωτήσεων Q1, Q2, Q3 και Q4 στο υλικό, να καταγράφουν τα αποτελέσματα και να συγκριθούν με αυτά των προσομοιώσεων.

- Να εκτελεστούν οι επερωτήσεις σε πολύ **μεγάλο όγκο δεδομένων** τόσο κατά την προσομοίωση όσο και κατά την υλοποίηση στο υλικό.

- Να καταγραφεί η **πραγματική ταχύτητα** των εκτελέσεων που γίνονται με μετρητές **απευθείας στο υλικό**.

- Να γίνει επέκταση των επερωτήσεων **πολύπλοκης μορφής**.

- Να γίνει χρήση επερωτήσεων με **περισσότερες παράλληλες επεξεργασίες**.

- Να γίνει χρήση **μεγαλύτερων μνημών** CAM και ROM- RAM.

- Να γίνει χρήση των **ενσωματωμένων επεξεργαστών** τόσο μαλακού πυρήνα (soft core) όσο και σκληρού (hard core), στην επεξεργασία επερωτήσεων.

- Να **συγκριθούν οι επιδώσεις** των συσκευών FPGA με αυτές των συμβατικών επεξεργαστικών μηχανών που χρησιμοποιούνται στην επεξεργασία δεδομένων.

- Να γίνει η **καταμέτρηση ισχύος** των συσκευών FPGA κατά την εκτέλεση των επερωτήσεων με αυτές των κοινών επεξεργαστικών μηχανών (CPU).

- Να **τροποποιηθεί ο αλγόριθμος CAM_ARRAY** ώστε να διαβάζει απευθείας από ένα αρχείο *.txt τον πατρικό πίνακα και να δημιουργεί απευθείας τα αρχεία αρχικοποίησης *.coe των μνημών CAM και RAM – ROM.

- Να δημιουργηθεί μία **"βιβλιοθήκη"** η οποία θα περιέχει όλα τα λογικά κυκλώματα που θα δημιουργούνται, με τις τεκμηριώσεις τους, ώστε να χρησιμοποιούνται για μελλοντική χρήση από άλλα λογικά κυκλώματα.

- **Χρήση άλλων θυρών επικοινωνίας** της συσκευής FPGA όπως Ethernet και PCI για τη διενέργεια πειραμάτων.

Υλοποίηση στο υλικό του FPGA της επερώτησης Q1

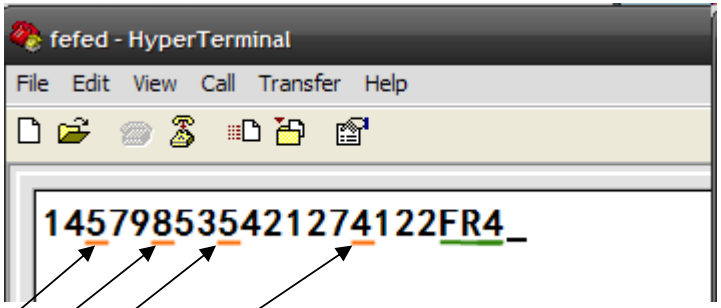
Q1: SELECT A_1, A_2, A_3
FROM R
WHERE $A_3 > '2'$

→

Q1: SELECT COUNT (*)
FROM R
WHERE $A_3 > '2'$

R

A_1	A_2	A_3
1	4	5
7	9	8
5	3	5
4	2	1
2	7	4
1	2	2



Q1

A_1	A_2	A_3
1	4	5
7	9	8
5	3	5
2	7	4

Για τον τερματισμό της εκτέλεσης θα πρέπει μέσα στο αρχείο *.txt (στο τέλος) να υπάρχουν τα κεφαλαία γράμματα FR.

Βιβλιογραφία

- **[Max] Clive "Max" Maxfield. «The Design Warrior's Guide to FPGAs». Mentor Graphics Corporation and Xilinx, Inc. Elsevier, ISBN 0-7506-7604-3, Copyright 2004.**
- **[KitS] Xilinx, MicroBlaze Development, Kit Spartan-3E 1600E, Edition User Guide**

Προγράμματα – Υλικό

Για την υλοποίηση της μεταπτυχιακής διατριβής χρησιμοποιήθηκαν τα παρακάτω προγράμματα:

- **Microsoft Office Word 2003**
- **Microsoft Office Excel 2003**
- **Microsoft Office Visio 2003**
- **Microsoft PowerPoint 2003**
- **Microsoft Hyper Terminal**
- **Xilinx ISE 10.1**
- **Eclipse**

Οι προσομοιώσεις έγιναν σε ηλεκτρονικό υπολογιστή με λειτουργικό σύστημα Microsoft Windows XP Professional Version 2002 Service Pack 3 και φέρει επεξεργαστή Intel Core2 CPU 4400@2.00GHz 2.01GHz, 1,97 GB RAM.

Η προσπάθεια υλοποίησης στο υλικό του FPGA έγινε στη συσκευή της Xilinx, Spartan-3E 1600E.

Ευχαριστώ!